Improving Reinforcement Learning Rates Using Prior Knowledge

KEITH SWANSON
ARTIFICIAL INTELLIGENCE RESEARCH BRANCH
NASA AMES RESEARCH CENTER
MAIL STOP 269-2
MOFFETT FIELD, CA 94035-1000

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA-93-25
August, 1993

Improving Reinforcement Learning Rates Using Prior Knowledge

Keith J. Swanson
swanson@ptolemy.arc.nasa.gov
Tel: 415-604-6016
NASA Ames Research Center
MS: 269-2, Moffett Field, CA 94035-1000, U.S.A.

Abstract

This paper presents a new technique called Prior Policy Fallback (PPF) which provides a simple means of incorporating prior knowledge into a reinforcement learning system based on Q-learning. PPF performs significantly better than an intuitively similar method based on selectively initializing values of the Q table. The benefits of PPF occur because PPF does not adversely interact with the normal Q-learning update mechanisms. PPF simply accelerates the time to reward during initial trials and then gradually becomes less involved as normal Q-learning updates occur. PPF provides an alternative to manual teaching methods for accelerating learning during early trials and can be used in conjunction with many of the existing methods for accelerating Q-learning updates. The benefits of PPF are illustrated through several experiments based on a static grid-based world.

1 Introduction

Engineered devices which interact with the physical world are often developed without the benefit of knowing exactly which situations the device may encounter during execution. When information about the expected operation environment is limited, it becomes difficult to define, in advance, appropriate responses for all expected situations. Reinforcement learning is an appealing technique for a device or *agent* to learn appropriate responses to situations as they are experienced. Reinforcement learning is reminiscent of our own trial-and-error learning periods of childhood where various possible actions were taken in many different situations, a few of which would result in reward (a parental "good job"). Eventually, we learn to construct sequences of actions which proactively lead us to reward.

The reinforcement learning problem can be loosely framed as one of learning a *policy* for selecting actions in various states of the world in order to achieve reward. A fairly popular, recent framework for formalizing the reinforcement learning problem is called Q-learning [Watkins, 1989]. Intuitively, Q-learning develops a policy by incrementally improving a Q-value assigned to every possible state-action pair of the environment. To execute the policy, we simply select the action with the largest Q-value for the current state. Q-learning is able to learn an optimal policy (subject to some conditions, [Watkins, 1989]) without any prior knowledge or model of the environment.

The problem with Q-learning, however, is that often learning rates are quite slow. Since rewards may only be received when a goal is fully satisfied (*e.g.*, consider a simple path planning task), the situations and actions must be experienced many times until the Q-values converge. To address the slow convergence problem, various techniques have been proposed which range from teaching the agent by providing sample execution traces [Lin, 1991], to building action models of the environment which are then used to develop hypothetical experiences to use in addition to actual experiences [McCallum, 92; Sutton, 1990]. A disadvantage of training methods is that they require a person to be involved during execution to provide the training. A disadvantage of methods based on hypothetical updates is that their benefit typically does not begin until after some reward has been initially achieved.

Incorporation of prior knowledge (*e.g.*, the goal is most likely to the east), is another technique which may be used to accelerate Q-learning rates. This paper presents a new technique called Prior Policy Fallback (PPF) which provides a simple means of incorporating prior knowledge into a Q-learning framework. The surprising benefit of PPF is that it can work much better than an intuitively similar method which involves modifying the initial Q-values. The next section of this paper provides a brief introduction to Q-learning. Section 3 presents the Prior Policy Fallback technique and section 4 provides experimental results which illustrate its benefit. Section 5 presents related work and Section 6 concludes.

2 Q-Learning

One set of techniques for determining an optimal policy is based on the theory of dynamic programming [Bertsekas, 1987]. Watkins [1989] has studied the relationship between dynamic programming and reinforcement learning and has proposed a new reinforcement learning technique called *Q-learning*. The Q-learning approach maintains, for each state-action pair, an estimate (Q-value) of the value of taking action a in state x. Q-value estimates are maintained for each state-action pair and are updated incrementally until the Q-values converge to optimal values. The Q-value measures the expected total discounted reward obtained by taking action a in state x, followed by taking actions per the optimal policy thereafter. That is,

$$Q(x,a) = E\{r + \gamma * V(y) \mid x, a\}$$

where γ, $0 \leq \gamma < 1$, is a temporal discount rate, r is the immediate reward obtained, and $V(y)$ is the maximum of the Q-values for the resulting state y.

Q-learning proceeds similarly to Sutton's [1988] method of temporal differences. An agent tries an action in a particular state, then evaluates the consequences of the action in terms of any immediate reward received *and* an estimate of the value of the state in which the agent finds itself. At each time step, the learning system observes the current state x, takes action a, observes the next state y, and receives immediate reward r. If we assume a tabular representation for the Q-values, $Q(x,a)$ will be left unchanged for all state-action pairs not equal to (x,a) and $Q(x,a)$ will be updated by

$$Q(x,a) = Q(x,a) + \alpha(r + \gamma * V(y) - Q(x,a))$$

where α, $0 < \alpha \leq 1$, is a learning rate parameter.

Watkins and Dayan [1992] show that, provided each state-action pair is experienced an infinite number of times, no matter what the initial Q-values are, they will eventually converge to the optimal policy. When all Q-values are initialized to zero, no Q-value updates occur until an action is taken which results in reward (*e.g.*, reaching a goal location). This causes the state which led to the reward to have its Q-value modified per the update equation given above. Thus after many experiences, Q-values will propagate backwards from the goal until an optimal gradient of Q-values is produced.

Q-learning systems actually employed on agents are subject to tension between "performing actions that are not well understood in order to gain information about their reinforcement value and performing actions that are expected to be good in order to increase overall reinforcement." [Kaebling, 1990] A typical approach is to include an exploration bonus b in the action selection algorithm for the agent. Rather that always choosing the action with the highest Q-value, b percent of the time an action is taken which is not a maximum Q-value

action. Sutton [1990] presents a more sophisticated exploration technique which modifies the Q-value update equation to cause unexplored state-action pairs to be actively sought out.

Initializing Q-values to zero is equivalent to defining an initial random policy. Under this policy, the agent will search (wander) randomly until it accidentally stumbles across a reward state and the first Q-value update is made. Whitehead [1991] states that "The time complexity of this search strongly depends upon the size and structure of the state space and upon *a priori* knowledge encoded in the learner's initial parameter values." He further provides a theorem which basically states that a zero-initialized Q-learning system can have a search time that is exponential in the depth of the state space. These results suggest that reward functions which do not yield any sort of gradient (*e.g.,* non-zero reward only at a goal location) provide insufficient feedback to allow tractable learning for complex domains.

In the following section we present a simple mechanism for encoding *a priori* knowledge which directly addresses the complexity associated with the initial searches in the domain.

3 Prior Policy Fallback

Prior Policy Fallback (PPF) is a simple mechanism for incorporating *a priori* knowledge about a domain *without* disrupting the existing Q-learning update mechanisms. The basic intuition is that when useful *a priori* information about a domain is available, it should be incorporated into the Q-learning framework. We will assume that the *a priori* knowledge can be represented as a function from a state x to a preferred action a to take in that state. The problem with simply trying to define an optimal policy in advance is that some information may not be available until the agent is actually deployed in the environment. If all required information was available, it would be straightforward to directly define the optimal policy and learning would not be required. More likely, knowledge is incomplete but some regularities about the domain are known in advance and these can be provided to the agent as part of a default or "prior" policy. Thus, defining a prior policy amounts to no more than simply defining a partial function from states to actions.

Once a prior policy has been defined, we can now slightly generalize the action selection mechanism of Q-learning. The standard model for action selection in Q-learning is that the Q-values for the state-action pairs associated with the current state are compared and the largest Q-value determines which action to select in the state. This scheme is often modified to include an exploration percentage as discussed above to occasionally select a non-maximum Q-value action. When all the Q-values associated with a state are identical (*e.g.,* when the system has been zero-initialized), we say the Q-values provide no advice.

We can now define Prior Policy Fallback as simply specifying that if the Q-values for a state provide no advice, then "fallback" on the prior policy to select an action. Thus, any *a priori* policy is clearly only considered until Q-value estimates have been propagated

to that state by the normal update mechanisms. The advantage of this approach is that the prior policy "gets out of the way" once Q-value updates start progressing. Note that an exploration percentage is still required because otherwise, the prior policy could be a deterministic policy that simply leads to a dead end.

Our basic goal in using PPF is to be able to define a prior policy, in combination with an exploration strategy, that will be more likely than a random policy to find the goal and start the normal Q-value updating. An alternative approach to requiring an exploration strategy would be to allow nondeterministic prior policies which have non-zero probabilities for all possible actions. We do not consider this approach further in this work.

One of the advantages of PPF is that it can be used in conjunction with existing techniques for accelerating the backwards propagation of Q-values ([Moore & Atkeson, 1993], [Peng & Williams, 1993]). Of course PPF is not without its limitations. The prior policy is just a heuristic and poorly chosen, it can hurt more than it helps.

4 Experiments

In this section we present the results from several experiments which compare PPF to a biased Q-value initialization approach. We also show that PPF can be productively used in conjunction with other existing techniques for accelerating Q-learning. The experiments presented in this section are modifications to several of the experiments presented in Sutton[1990] regarding the DYNA-Q system.

DYNA is an architecture for exploring tradeoffs between systems that plan using an internal world model, systems that react using precompiled situation-action rules, and systems that learn through some type of trial-and-error or reinforcement learning. Sutton introduced a simple navigation task to evaluate the performance of his DYNA systems. Although Sutton used this specific task to evaluate the performance of his DYNA-PI (policy iteration) system, we use it here to evaluate Q-learning approaches. The task involves navigation of an agent in a grid-based world from a starting location to a goal location. We have represented the agent's world using a simulator called NASA TileWorld [Philips & Bresina, 1991]. Obstacles are present in the world as seen in Figure 1. The agent's initial location is indicated by the small circular icon near the west (left) edge of the world. The goal location is not indicated but is the square in the north-east corner of the world.

The learning task is to develop a policy which will direct the agent over the shortest path from the initial starting location to the goal location. The reward system gives the agent a reward of one for reaching the goal location and zero for all other states. We define a *trial* to be one successful trip from the starting location to the goal location. After completing a trial and receiving reward, the agent is automatically transferred back to the starting location to begin a new trial. If we assume that the agent is given no prior knowledge (zero-initialized Q-values), the initial behavior of the agent will be that of a random walk. The initial trials

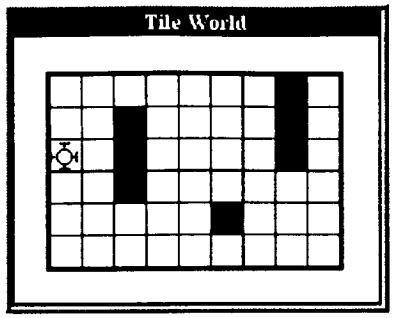

Figure 1: Test Environment

may take many steps because the expected number of steps required to "stumble" across the goal will be quite large. However, as trials are completed, Q-value updates begin to propagate Q-values back from the goal location and the steps per trial will decrease until finally reaching the shortest path length.

Following Sutton [1990], system performance is measured as the number of steps required per trial. For this series of experiments, the learning rate and the discount rate were simply chosen from the values selected for Sutton's DYNA-Q experiments [Sutton, 1990]. The learning rate α was set at 0.5 and the temporal discount rate γ was set at 0.9. The exploration percentage was set so that 10% of the time, an exploration action would be taken. Note that these experiments assume a deterministic world. Issues associated with actually learning the action models for the environment are not part of this work. As in Sutton [1990], we simply assume that an action model will be learned. Based on this action model, we have only included the possible state-action pairs in the tabular representation of Q-values.

To evaluate the relative benefits of PPF, a random walk (zero-initialized Q-values) was chosen as the baseline. Against this baseline we wish to compare two intuitively similar techniques for encoding prior knowledge. The prior knowledge we wish to encode will simply be the heuristic "go east". The first technique, called Prior Initialized Q-values, will simply involve initializing the Q-values to a non-zero value for all east moves. For this experiment, an initial Q-value of 1.0 was arbitrarily chosen for these east moves. The second approach for encoding the "go east" heuristic knowledge is based on PPF. A simple function was written which given a state, returns a move of east if it is possible in that state.

To perform action selection using PPF, the current Q-values for the state, the exploration percentage, and the default policy may all be involved. If only one action is possible in a state, this action will be selected. If there are any actions other than those having maximum Q-value, the exploration strategy will cause one of those actions to be chosen randomly, 10% of the time. If we have uniform Q-values, and a prior policy (per PPF) which is recommending an action, the prior policy action will be selected. If none of the preceding conditions hold, then one of the Q-value recommended actions is chosen at random (since there may more than one action having the same maximum Q-value).

The results of this experiment are shown in Figure 2. The vertical axis shows the number

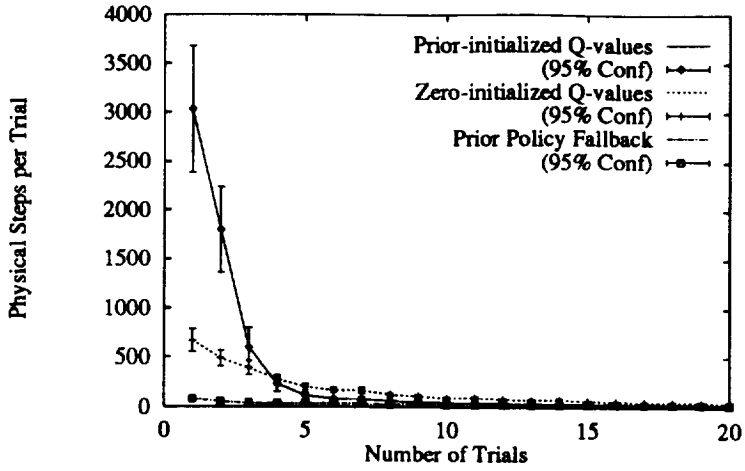

Figure 2: Comparison of Prior Knowledge Encodings

of physical steps required per trial. The horizontal axis indicates the trial number. Three curves are shown corresponding to the prior-initialized case, the zero-initialized case, and the PPF case. For each curve, error bars indicating a 95% confidence interval are also shown.[1] To facilitate identification of the curves using the key, the vertical order of the key entries corresponds to the vertical ordering of the leftmost point of each curve.[2] As expected, the number of steps per trial decreases as the agent gains more experience in the world (more trials). We see that the zero-initialized Q-learning system takes about 600 steps for the initial trial and gradually improves. We also see that, as we had hoped, PPF provides a significant improvement over the zero-initialized case. For the prior-initialized case we have a somewhat surprising result. Although after 4 trials it does better than the zero-initialized case, initially it does much worse, taking several thousand steps during the early trials. This result is somewhat surprising because prior-initialized Q-values and PPF both seem to be intuitively similar methods for encoding our "go east" heuristic, but they clearly perform very differently.

The results shown in Figure 2 raise an obvious question about the potential impact of the magnitude of the Q-value chosen for prior-initialization. The Figure 2 results were obtained with an arbitrarily chosen Q-value of 1.0 for all east moves. Considering that Q-values eventually converge to a gradient, might a different choice for initial Q-values do substantially better? Intuitively, initial Q-values close to the resulting gradient values would be more desirable. However, chosing those values is more difficult than encoding a static "go-east" bias. Further, those values may not transfer across problems. For example, if the obstacles of the DYNA world were rearranged, the Q-value gradient could be quite different. What we wish to encode here is a simple bias that can be useful over many different problem instances.

To evaluate the impact of the prior-initialization Q-value, two additional tests were run with initial east Q-values of 0.01 and 100. These results, along with the original 1.0 prior-initialization results, are shown in Figure 3. Error bars have been removed to make it

[1] Typically 50 to 100 samples were taken per curve for all figures in this paper.

[2] This type of key ordering has been used for each of the plots in this paper.

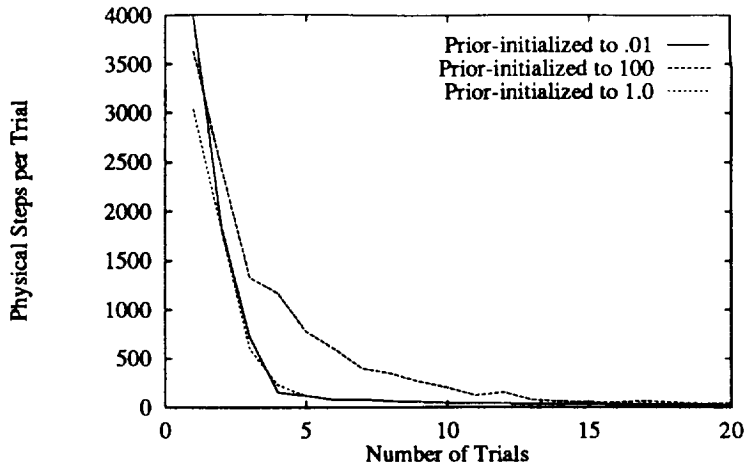

Figure 3: Comparison of Initial Q-value Settings

easier to distinguish the lines. We see that these other Q-values are actually worse than our initial guess of 1.0. Although we may be able to further improve on the results obtained with initialization at 1.0, multiple repetitive attempts are likely to be required unless we actually attempt to compute the Q-values for convergence. Of course if we do this then we've essentially solved the problem. Thus, these results help confirm that trying to choose initial Q-values to encode an heuristic bias is not as effective as PPF for accelerating learning rates during initial trials.

A final experiment was conducted to compare the benefits of PPF with the form of incremental hypothetical planning explored by Sutton [1990]. For this final experiment, Sutton's hypothetical experience mechanism was implemented and the same learning rate and discount rate parameters were used as in the previous experiments. The only difference is that after every physical action, it was possible for the agent to experience a number of hypothetical actions based on the learned world model. Each hypothetical action was chosen randomly from among those previously experienced. As Sutton[1990] demonstrated, combining hypothetical actions with physical actions can significantly improve the rate at which Q-learning converges. Techniques other than random choice ([Moore & Akinson, 1993], [Peng & Williams, 1993]) can improve the rate even more.

The results of the comparison between PPF and DYNA-style planning are shown in Figure 4. All the curves are based on Q-values that have been zero-initialized. Ten planning steps were experienced for each physical action. We see that PPF performs better than the ten step planning case for the early trials but that the planning approach converges more quickly to optimal values. Also included is one curve which is a combination of both PPF and ten step planning. This approach provides the fastest convergence. A key benefit of PPF is that it can be used in combination with existing techniques for accelerating Q-value propagation. Figure 4 confirms that PPF, in conjunction with planning, improves upon the results obtained by planning alone.

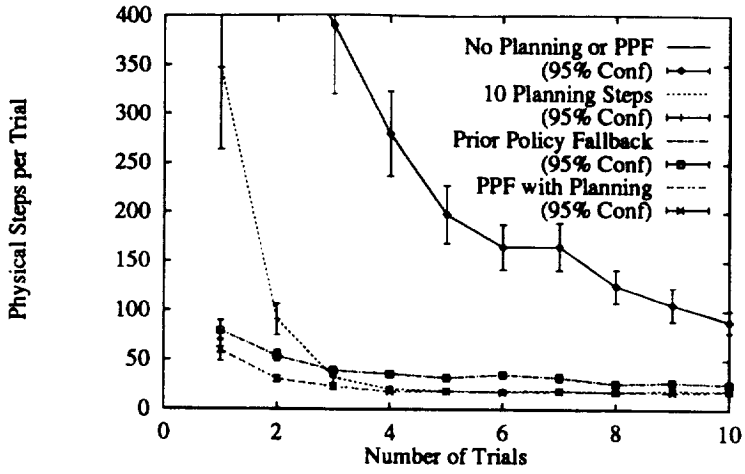

Figure 4: Comparison of Planning and Prior Policy Fallback

5 Related Work

Slow learning rates are only one of the limitations of Q-learning. Formal convergence results of Watkins and Dayan [1992] only hold when Q-values are represented as a table; for any practical domain, the state space will be too large to represent all state-actions pairs explicitly so some form of state generalization will be required. Lin [1990] has reported good results by combining a connectionist backpropagation mechanism [Rumelhart, 1986] for input generalization with reinforcement learning techniques. Chapman and Kaelbling [1991] present an algorithm which incrementally builds a tree-structured Q-table similar in some respects to induced decision trees [Quinlan, 1986].

If an action model for the environment can be learned, a number of methods related to heuristic search and dynamic programming can be used to exploit the action model. Several techniques have been explored which attempt to prioritize the hypothetical experiences that can be used to accelerate the propagation of the Q-values. Peng & Williams [1993] present a technique called Queue-DYNA in which Q-value updates are prioritized and only those having the highest priority are performed at each time step. Moore & Atkeson [1993] present a similar technique they call Prioritized Sweeping. McCallum [1992] presents a related method called Transitional Proximity Q-learning. These techniques tend to be based on heuristic methods for focusing efforts on "interesting" parts of the Q-table.

Prioritized Sweeping and Queue-DYNA both tend to prefer Q-value updates that spread out backward from the goal state. Korf [1990] provides an algorithm called LRTA* which works outward from the current state and Barto *et al.* [1993] generalize Korf's algorithm and relate it to Q-learning methods. Mahadevan [1992] infers action models using a statistical clustering technique and then uses these models to perform a multi-step lookahead to help select the most appropriate action. Mahadevan's work also provides an example of a technique which assumes a reward gradient is available to speed up learning during the initial trials. Once a learned world model is available, a large body of results from the search and planning literature potentially become relevant and work is underway to exploit these

relationships. Singh [1992] provides one example where hierarchical planning methods are being integrated with reinforcement learning methods.

Teaching methods are appealing because, unlike the previous methods, they help to reduce the time required until initial reward is achieved. Whitehead [1991] provides results which emphasize that lack of *a priori* knowledge can make the search during initial trials intractable. He proposes a teaching method called "Learning with an External Critic" where a mentor watches the learner and generates immediate rewards in response to its most recent actions. He also presents a method, "Learning by Watching", where an agent gains experience vicariously by observing the behaviors of other agents. Other teaching methods are possible. Clouse & Utgoff [1992] present a method that allows a teacher to occasionally suggest an action when it appears the agent is not progressing well. Lin [1991] presents a technique called experience replay, which can be used in conjunction with teaching, that collects a sequence of physical experiences and then replays them backwards to help accelerate the propagation of Q-values. The disadvantage of teaching methods is that they require online involvement of the mentor. Teaching methods do not directly provide a means for specifying simple heuristic knowledge (*e.g.*, "go east") before the agent begins experiencing the environment.

This work has been motivated by the work of Drummond *et al.* [1993] where a programmer-provided policy is used to bias search to areas more likely to contain solutions. Finally, there has been other work which has also attempted to integrate prior knowledge within a reinforcement learning framework. Berenji [1992] represents *a priori* knowledge in the form of fuzzy control rules and uses reinforcement learning methods to improve the performance of these control rules. Franklin [1987] represents *a priori* knowledge as a conventionally designed fixed controller and then uses reinforcement learning to improve the performance of the controller.

6 Conclusions and Future Work

A range of approaches are being developed which accelerate Q-learning rates. However, most of these approaches do nothing to reduce the time required until initial reinforcement is received. If nothing but a random initial policy is available, time to initial reward can be exponential in the solution length. Prior Policy Feedback (PPF) directly addresses this problem by providing a simple means of encoding *a priori* knowledge that does not adversely interact with the normal Q-learning update mechanisms. PPF has been experimentally shown to perform significantly better than intuitively similar methods that involve initializing selected Q-values in the Q-value table. A further advantage of PPF is its ability to be used in combination with other methods that heuristically prioritize Q-value updates to speed the propagation of Q-values.

PPF works well when the *a priori* knowledge to be encoded is static. However, there will be cases when heuristics need to be time-varying, perhaps because the reward structure is

changing. Since the prior policy of PPF is ignored once Q-values have begun to propagate throughout the space, there is no obvious way to reuse the advice of the prior policy, even if it becomes advantageous to do so. Further work is needed to attempt to extend the applicability of PPF to more complex situations. Additionally, PPF may also be useful for other non Q-learning reinforcement learning approaches but these connections have not yet been pursued. As Kaebling [1990] concludes in her dissertation, "Finally, we might start from a complete or partial program specified in terms of condition-action rules. An interesting research direction would be to develop representations of programs that are amenable to adjustment using reinforcement-learning methods." Although PPF policies are not directly modified by reinforcement learning methods, the encouraging results obtained with PPF suggest further work in this direction will be fruitful.

Acknowledgements

Thanks to Anton Schwartz for a stimulating lecture on reinforcement learning that got me interested in this topic. Anton and Phil Laird provided several clarifying discussions. Also thanks to Anton for several paper pointers and to Hamid Berenji for letting me borrow his pile of reinforcement learning papers that greatly facilitated my initial survey of the field. Finally, thanks to Hamid, John Bresina, and Mark Drummond for providing comments on a draft of this paper.

References

[1] Barto, A., Bradtke, S., & Singh, S. 1993. Learning to Act using Real-Time Dynamic Programming. Submitted to *AI Journal special issue on Computational Theories of Interaction and Agency.*

[2] Berenji, H. 1992. Learning and Tuning Fuzzy Logic Controllers Through Reinforcements. *IEEE Transactions on Neural Networks*, Vol. 3, No. 5.

[3] Bertsekas, D. 1987. Dynamic Programming: Deterministic and Stochastic Models. Englewood, Cliffs, NJ: Prentice Hall.

[4] Chapman, D., & Kaelbling, L. 1991. Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons. *Proceedings of the 12th International Joint Conference on Artificial Intelligence.*

[5] Clouse, J., & Utgoff, P. 1992. A Teaching Method for Reinforcement Learning. *Proceedings of the Ninth International Workshop on Machine Learning.*

[6] Drummond, M., Swanson, K., Bresina, J., & Levinson, R. 1993. Reaction-First Search. *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, to appear.

[7] Franklin, J. 1988. Refinement of Robot Motor Skills Through Reinforcement Learning. *Proceedings of the 27th IEEE Conference on Decision and Control*, Austin Texas.

[8] Kaebling, L. 1990. Learning in Embedded Systems. PhD thesis, Stanford University.

[9] Korf, R. 1990. Real-Time Heuristic Search. *Artificial Intelligence* **42**:189-211.

[10] Lin, L. 1991. Programming Robots Using Reinforcement Learning and Teaching. *Proceedings of the Ninth National Conference on Artificial Intelligence.*

[11] Mahadevan, S. 1992. Enhancing Transfer in Reinforcement Learning by Building Stochastic Models of Robot Actions. *Proceedings of the Ninth International Workshop on Machine Learning.*

[12] McCallum, R. 1992. Using Transitional Proximity for Faster Reinforcement Learning. *Proceedings of the Ninth International Workshop on Machine Learning.*

[13] Moore, A., & Atkeson, C. 1993. Prioritized Sweeping: Reinforcement Learning with Less Data and Less Real Time. *Machine Learning*, to appear.

[14] Peng, J., & Williams, R. 1993. Efficient Learning and Planning Within the DYNA Framework. In *Proceeding of the Second International Conference on Simulation of Adaptive Behavior*, Honolulu, HI. To appear.

[15] Philips, A., & Bresina, J. 1991. *NASA TileWorld Manual* NASA Technical Report TR-FIA-91-11. Moffett Field, CA: NASA Ames Research Center, Code FIA.

[16] Rumelhart, D., Hinton, G., & Williams, R. 1986. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing* (Vol. 1) Cambridge, MA: MIT Press.

[17] Singh, S. 1992. Reinforcement Learning with a Hierarchy of Abstract Models. *Proceedings of the Tenth National Conference on Artificial Intelligence.*

[18] Sutton, R. 1990. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. *Proceedings of the Seventh International Conference on Machine Learning.*

[19] Quinlan, J. 1986. Induction of Decision Trees. *Machine Learning*, **1**:81-106.

[20] Watkins, C. 1989. Learning From Delayed Rewards. PhD thesis, Cambridge University, Cambridge, England, 1989.

[21] Watkins, C. & Dayan, P. 1992. Q-Learning. Technical note in *Machine Learning* **8**: 279-292.

[22] Whitehead, S. 1991. A Complexity Analysis of Cooperative Mechanisms in Reinforcement Learning. *Proceedings of the Ninth National Conference on Artificial Intelligence.*